**Programando Web Services com SOAP**

**Capítulo 4. O Web Service Publisher**

O web service Publisher é uma demonstração de um serviço web mais complexo modelado a partir daquele usado pelo SOAP Web Services Resource Center (<http://www.soap-wrc.com/>).  
Este serviço demonstra técnicas para implementar formas mais complicadas de web services. Ele se baseia no exemplo Hello World do Capítulo 3.

**4.1 Visão Geral**

O web service Publisher gerencia um banco de dados de itens de notícias importantes, artigos e recursos que se relacionam ao SOAP e a web services em geral.

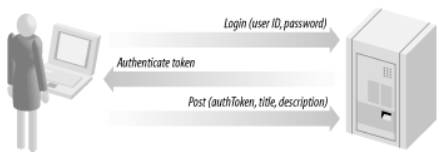
Um serviço baseado em Perl permite que usuários registrados postem, excluam ou naveguem por itens, e gerenciem suas informações de registro. Também implementamos um cliente shell interativo em Java que usa o cliente Apache SOAP.

As operações suportadas são:

* **register** – Criar uma nova conta de usuário.
* **modify** – Modificar uma conta de usuário.
* **login** – Iniciar uma sessão de usuário.
* **post** – Postar um novo item no banco de dados.
* **remove** – Remover um item do banco de dados.
* **browse** – Navegar no banco de dados por tipo de item. Os dados podem ser retornados em um formato XML específico do Publisher ou como um canal Rich Site Summary (RSS).

**4.1.1 Segurança do Serviço Publisher**

A segurança no serviço Publisher é tratada por meio de uma operação de login que retorna um token de autorização para o usuário.  
Este token consiste em um ID de usuário, endereço de e-mail, hora do login e um resumo MD5 que o usuário deve incluir em todas as operações que requerem autenticação — especificamente, as operações **post** e **remove** (veja Figura 4-1).



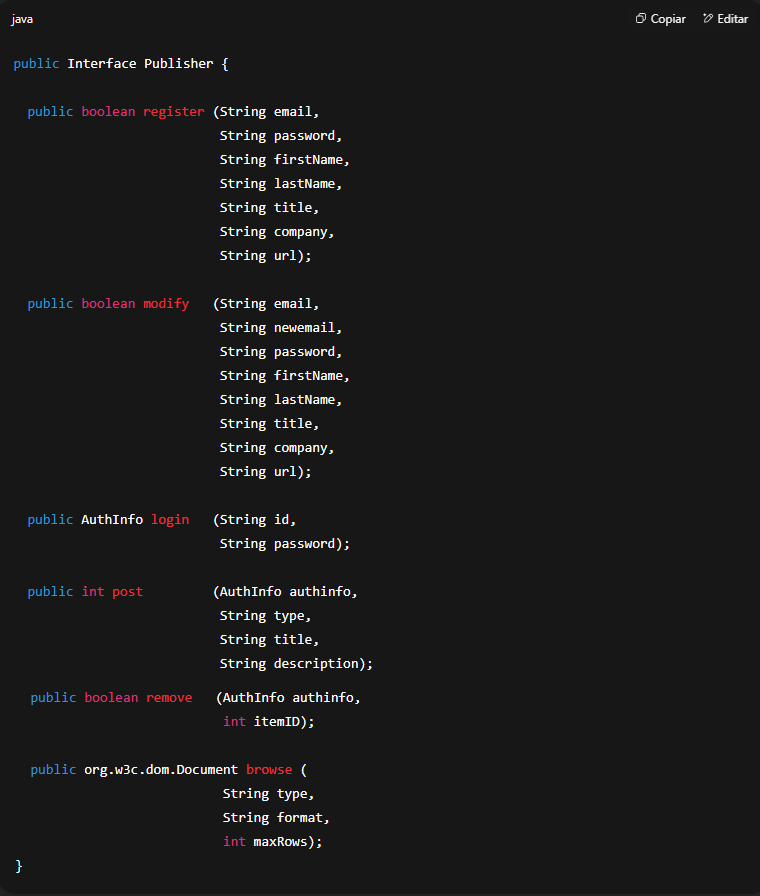
Na operação de login, o ID e a senha do usuário são enviados (em texto puro) para o serviço Publisher, onde são validados.  
O serviço então cria um token de autenticação e o retorna ao usuário.

Embora não seja muito seguro, isso ilustra uma forma de autenticação em web services baseados em SOAP.  
Em vez de usar mecanismos de segurança no nível de transporte, como autenticação HTTP, a segurança pode ser incorporada diretamente à interface do web service.

No Capítulo 5, discutiremos mecanismos de segurança muito mais robustos e seguros para web services.

**4.2 As Operações do Publisher**

As operações expostas pelo serviço Publisher são bastante diretas.  
Se modelássemos essas operações em uma interface Java, elas se pareceriam com o **Exemplo 4-1**:



**4.3 O Servidor Publisher**

O módulo Publisher em Perl usa o pacote **Perl DBI** e o pacote **DBD::CSV**, ambos disponíveis no CPAN e instalados da mesma forma que o **SOAP::Lite**.  
O código discutido a seguir deve estar contido em um único módulo Perl chamado **Publisher.pm**, mostrado integralmente no Apêndice C.

O código é bastante simples:  
Criamos um banco de dados para armazenar notícias, artigos e recursos, bem como a lista de usuários que utilizarão o serviço.  
Depois que o banco de dados é criado, definimos as operações para manipulá-lo. Essas operações não são exportadas.

A última parte do script gerencia logins e expõe as operações que o web service suportará.

**4.3.1 O Preâmbulo**

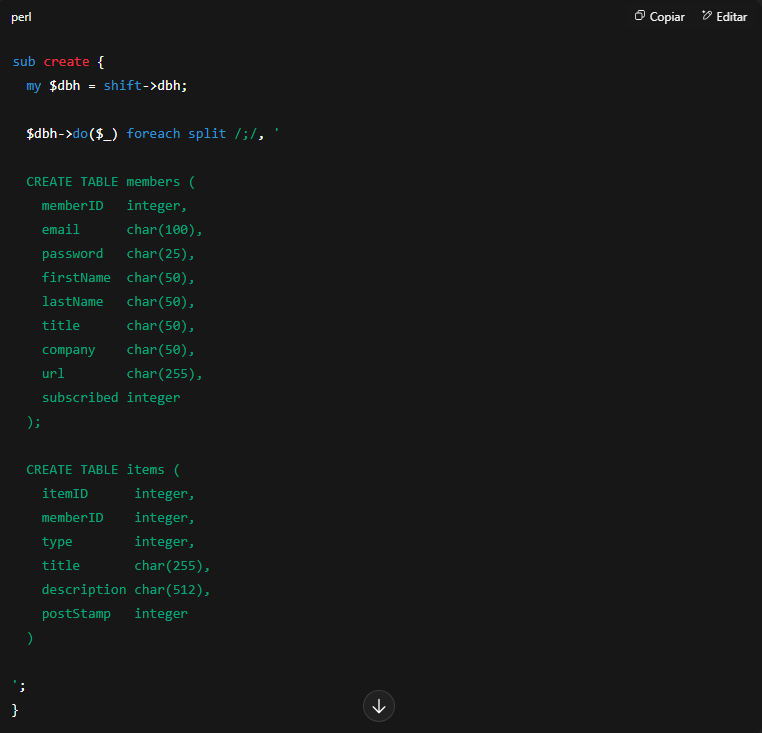
O **Exemplo 4-2** define o namespace do código, carrega o módulo do banco de dados e define uma função de conveniência para acessar o manipulador do banco de dados.

Os dados são armazenados em um arquivo texto separado por vírgulas, mas é possível trocar para um banco relacional, como MySQL, alterando a string "DBI:CSV:..." para o especificador de fonte de dados do novo banco.



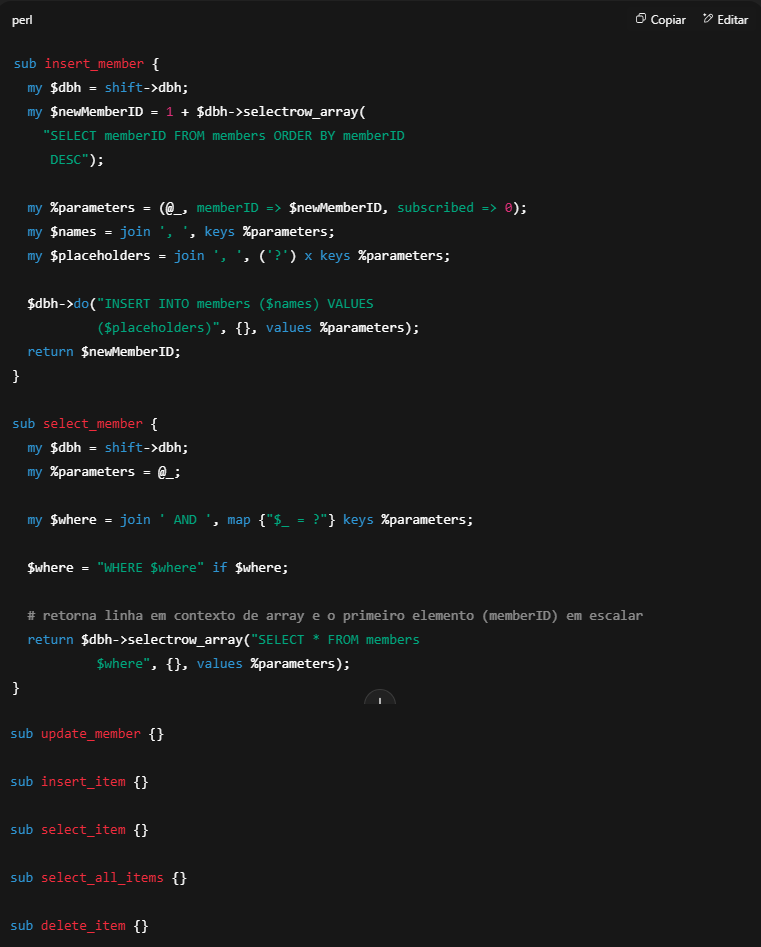
**4.3.2 Tabelas de Dados**

O **Exemplo 4-3** cria as tabelas para armazenar informações sobre membros e itens gerenciados pelo serviço Publisher.



Uma vez que as tabelas são criadas, precisamos escrever o código para manipular os dados nelas.  
Esses métodos, mostrados no **Exemplo 4-4**, são privados e **não** serão expostos como parte do nosso web service. Apenas os primeiros métodos são mostrados na íntegra. Consulte o Apêndice C para o código-fonte completo.

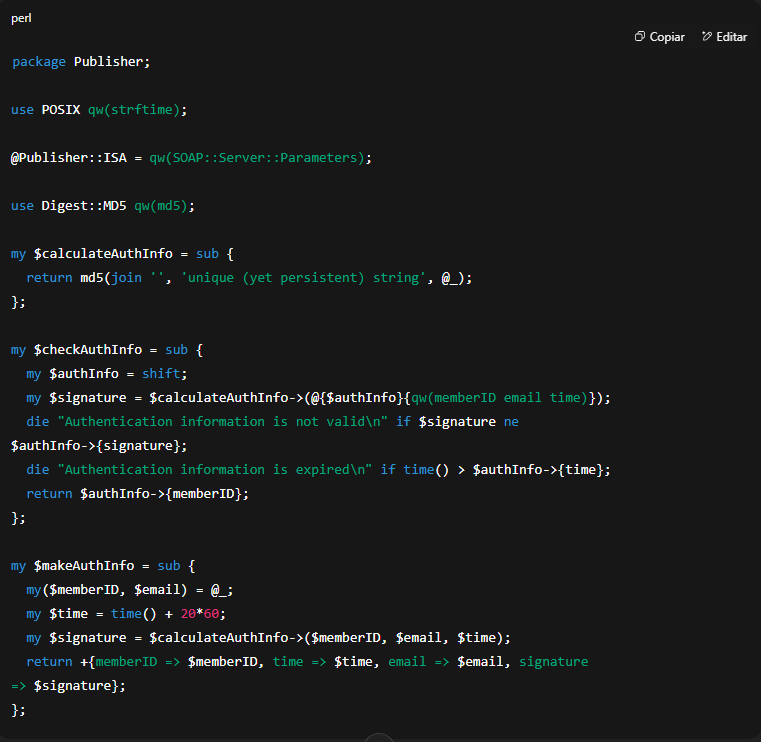
**Exemplo 4-4. Métodos para manipular dados nas tabelas**



**4.3.3 Funções Utilitárias**

Agora começamos a definir o web service Publisher propriamente dito.  
O **Exemplo 4-5** mostra várias funções utilitárias privadas, principalmente para lidar com a criação e validação dos tokens de autorização usados como parte do modelo de segurança do Publisher (discutido anteriormente).

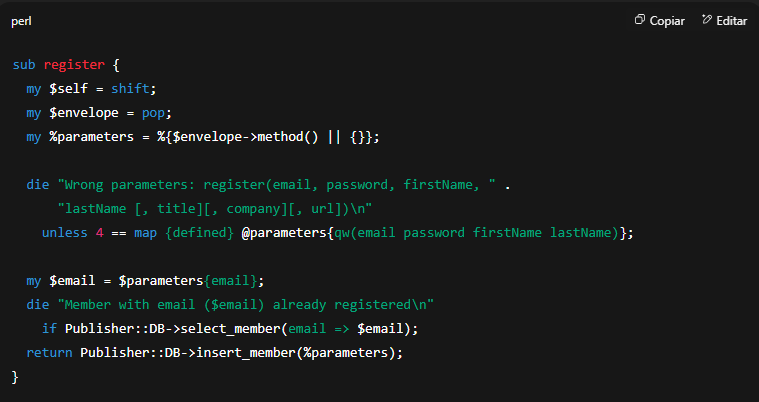
**Exemplo 4-5. Funções utilitárias**



**4.3.4 Registrar um Novo Usuário**

O **Exemplo 4-6** mostra o código para a operação exportada que registra novos usuários.

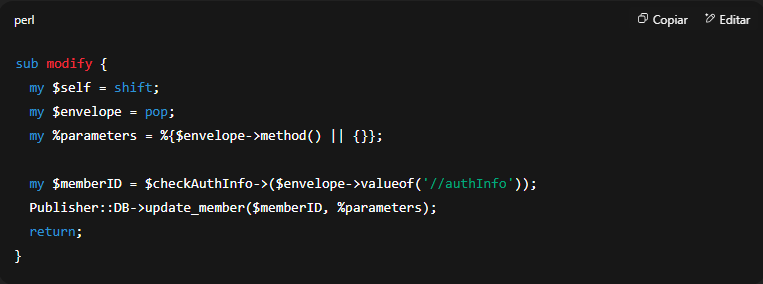
**Exemplo 4-6. Método exportado para registrar um novo usuário**



**4.3.5 Modificar Informações do Usuário**

O **Exemplo 4-7** é a operação que permite aos usuários modificar suas informações.

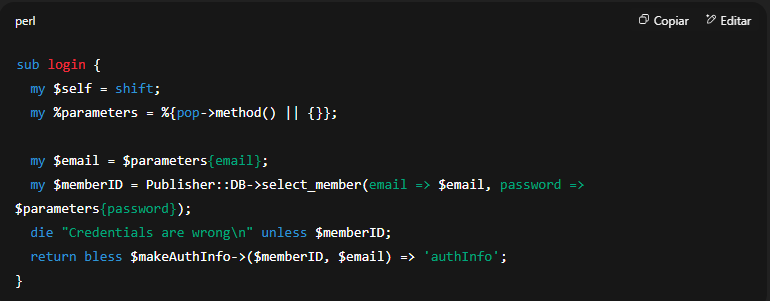
**Exemplo 4-7. Subrotina exportada para modificar informações de um usuário**



**4.3.6 Login de Usuário**

O **Exemplo 4-8** é a operação que valida o ID e a senha de um usuário e emite um token de autenticação.

**Exemplo 4-8. Método exportado para validar um usuário e emitir um token**



**4.3.7 Postando um Item**

O **Exemplo 4-9** mostra o método que posta um novo item no banco de dados.

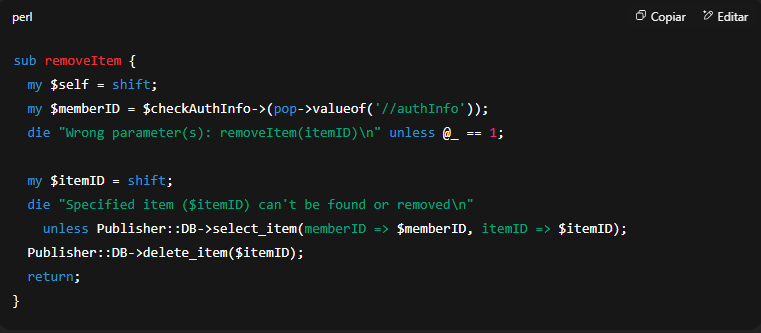
**Exemplo 4-9. Método exportado para postar um novo item**



**4.3.8 Removendo Itens**

O **Exemplo 4-10** mostra o método exportado para remover itens do banco de dados.  
Somente o usuário que adicionou um item pode removê-lo.

**Exemplo 4-10. Método exportado para remover um item do banco de dados**

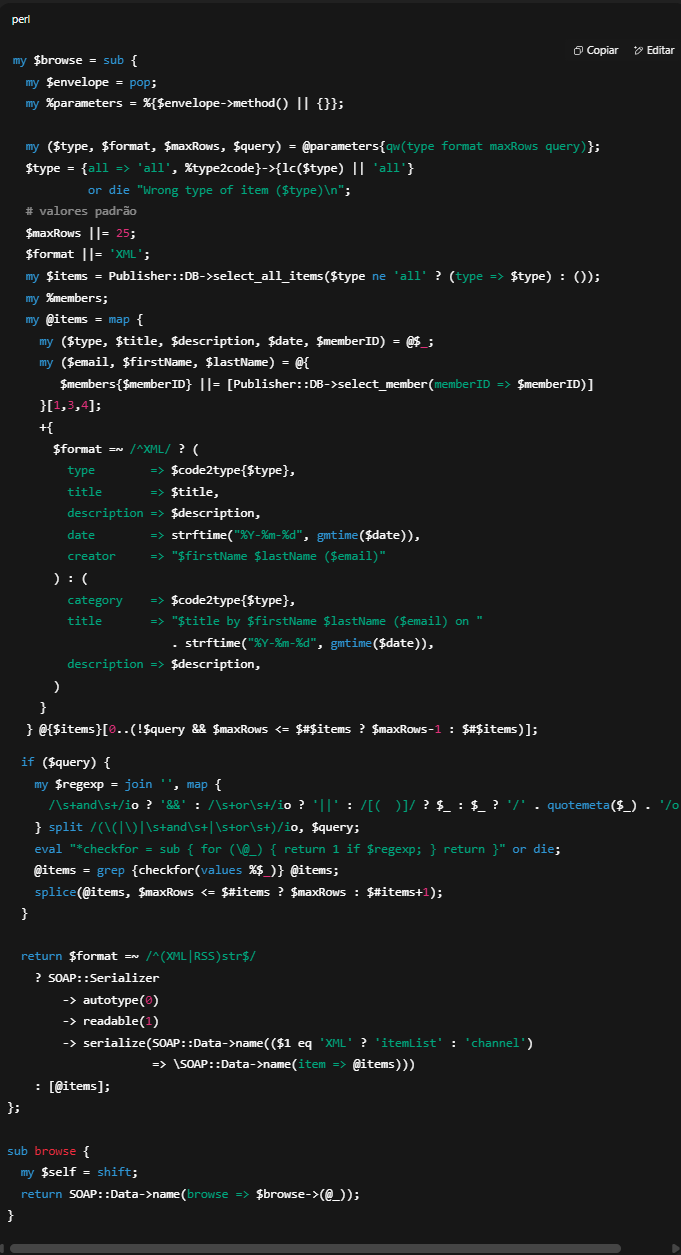


**4.3.9 Navegação**

Os usuários podem navegar no banco de dados de itens usando um formato XML específico do serviço Publisher ou o formato popular RSS (Rich Site Summary) amplamente usado na Internet.

O **Exemplo 4-11**, embora pareça bastante complexo, cria as estruturas XML apropriadas dependendo do formato solicitado pelo chamador.

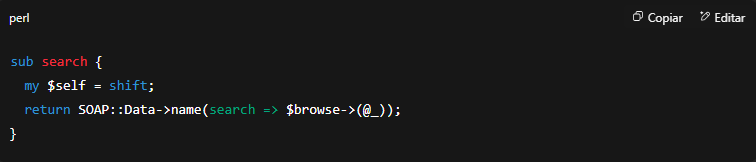
**Exemplo 4-11. Código para suportar navegação nos formatos proprietário e RSS**



**4.3.10 Busca**

A operação de busca é semelhante à operação de navegação, com a exceção de que os usuários podem especificar um filtro de palavra-chave para limitar o número de itens retornados.  
Está mostrada no **Exemplo 4-12**.

**Exemplo 4-12. Método exportado para buscar no banco de dados**



**4.3.11 Implantando o Serviço Publisher**

Para implantar o serviço Publisher, você precisa fazer duas coisas.  
Primeiro, criar o banco de dados que armazenará as informações.  
Faça isso executando o script mostrado no **Exemplo 4-13**.

**Exemplo 4-13. Programa para criar o banco de dados**



Isso criará dois arquivos no diretório atual, chamados **members** e **items**.

Em seguida, crie o script CGI que escutará mensagens SOAP e as encaminhará para o **SOAP::Lite** e o módulo **Publisher**.  
Este script está no **Exemplo 4-14**.

**Exemplo 4-14. Publisher.cgi, proxy SOAP para o módulo Publisher**



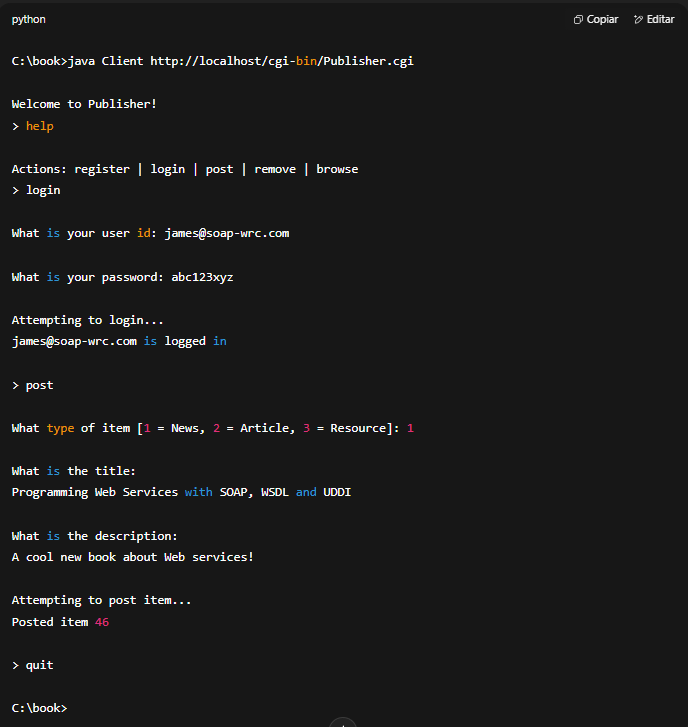
A chamada ao método **dispatch\_to** instrui o pacote **SOAP::Lite** sobre quais métodos aceitar e em qual módulo esses métodos podem ser encontrados.

Copie o script CGI para o diretório **cgi-bin** do seu servidor web e instale os arquivos **Publisher.pm**, **members** e **items** no diretório de módulos Perl.  
O web service Publisher está agora pronto para uso.

**4.4 O Cliente Shell em Java**

O cliente shell em Java é uma interface simples para interagir com o web service Publisher.  
Uma sessão típica é mostrada no **Exemplo 4-15**.  
Observe que, uma vez iniciado o shell, o usuário deve fazer login antes de postar novos itens.

**Exemplo 4-15. Uma sessão de exemplo com o cliente shell em Java**

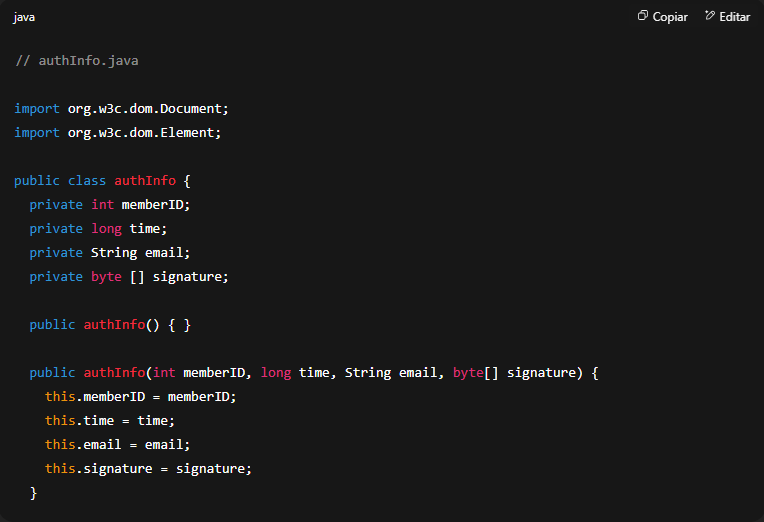


Para criar o shell, você precisa criar duas classes Java:  
uma para o próprio shell (**Client.java**) e outra para manter o token de autorização emitido pelo serviço Publisher quando você faz login (**AuthInfo.java**).

**4.4.1 A Classe de Autenticação**

O preâmbulo da classe **authInfo** é mostrado no **Exemplo 4-16**.

**Exemplo 4-16. A classe authInfo**



A classe possui os métodos **get** e **set** usuais.  
O **Exemplo 4-17** mostra os quatro primeiros métodos e esboça o restante.  
Para o código-fonte completo, veja o Apêndice C.

**Exemplo 4-17. Métodos de acesso da classe authInfo**



O método **serialize** cria uma representação XML de uma instância da classe **authInfo** que se parece com o **Exemplo 4-18**.

**Exemplo 4-18. Exemplo de serialização da classe authInfo**



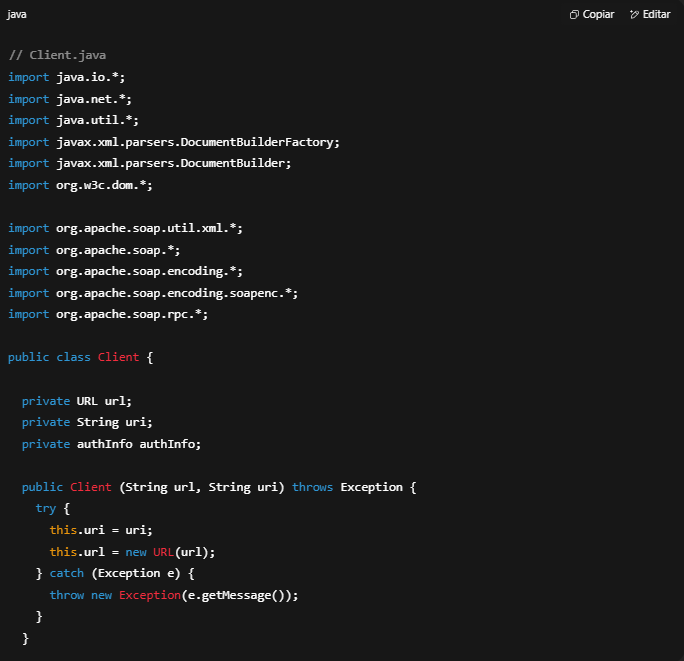
**4.4.2 A Classe Client**

A classe **Client** é direta.  
Existem rotinas utilitárias para trabalhar com o objeto cliente SOAP, algum código para lidar com autenticação e login, métodos para fazer chamadas SOAP para cada uma das operações que o usuário pode desejar executar e, por fim, um método **main** para lidar com a interface com o usuário.

**4.4.2.1 Preâmbulo**

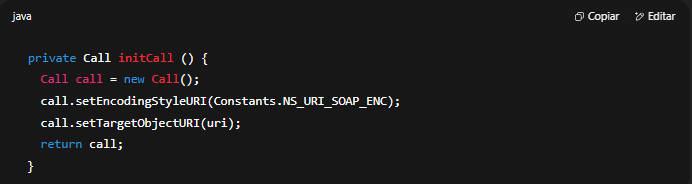
O preâmbulo da classe **Client** é mostrado no **Exemplo 4-19**.

**Exemplo 4-19. A classe Client**



O método **initCall** no **Exemplo 4-20** inicializa o cliente Apache SOAP.

**Exemplo 4-20. O método initCall**



O método **invokeCall** mostrado no **Exemplo 4-21** faz as chamadas para o serviço Publisher.  
Isso é semelhante ao exemplo do serviço Hello World que fornecemos anteriormente.

**Exemplo 4-21. O método invokeCall**

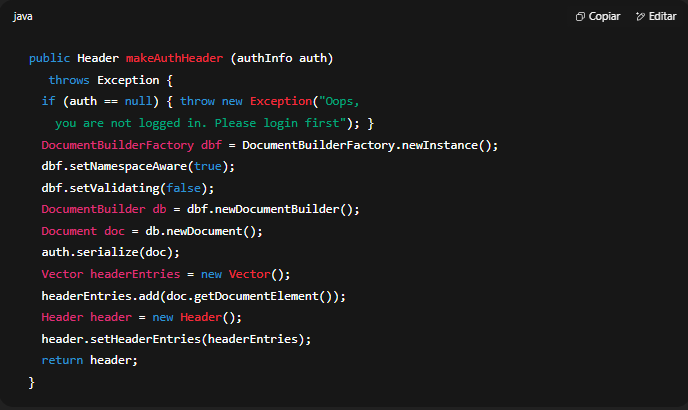


**4.4.2.2 Autenticação**

A operação **makeAuthHeader** no **Exemplo 4-22** cria um bloco de cabeçalho SOAP que contém um token de autenticação.  
Essa operação deve ser chamada sempre que alguém quiser postar ou remover itens no serviço Publisher.

Ela funciona simplesmente criando um documento DOM, instruindo a classe **authInfo** a se serializar nesse documento (veja o método **serialize** na classe **authInfo** no Exemplo 4-18) e adicionando as informações de autenticação aos cabeçalhos.

**Exemplo 4-22. O método makeAuthHeader**



**4.4.2.3 Login do Usuário**

O **Exemplo 4-23** mostra a operação de login.  
Observe que antes de invocar a requisição, devemos informar ao Apache SOAP qual desserializador usar para o token de autenticação retornado, caso a operação seja bem-sucedida.

O **BeanSerializer** é uma classe utilitária que vem com o Apache SOAP para traduzir XML em instâncias de classes Java que seguem o padrão Java Bean.  
Devemos informar explicitamente ao Apache SOAP que queremos que todos os elementos XML **authInfo** encontrados em uma mensagem SOAP dentro do namespace [**http://www.soaplite.com/Publisher**](http://www.soaplite.com/Publisher) sejam desserializados usando a classe **BeanSerializer**.

Se não fizermos isso, ocorrerá um erro sempre que um elemento **authInfo** for encontrado no envelope SOAP.

Falamos anteriormente sobre mapeamentos de tipos (**type mappings**) no Apache SOAP, mas ainda não explicamos como funcionam.  
Um mapeamento de tipo é um vínculo entre algum tipo de dado nativo (como uma classe Java) e a forma como esse tipo aparece como XML.  
Serializadores e desserializadores são códigos especiais capazes de traduzir entre os dois.  
O **SOAPMappingRegistry** é uma coleção de todos os mapeamentos de tipos e seus respectivos serializadores e desserializadores.

No Apache SOAP, devemos declarar um mapeamento de tipo sempre que quisermos usar qualquer tipo de dado que não seja um tipo primitivo incorporado (por exemplo, strings, inteiros, floats etc.).

**Exemplo 4-23. O método login**



**4.4.2.4 Wrappers para chamar as operações remotas**

Embora o cliente shell tenha métodos para cada uma das operações do web service Publisher, isso não é estritamente necessário.  
Fizemos dessa forma neste exemplo para garantir que você tenha uma visão clara de como o envelope SOAP é construído e utilizado.

Isso seria mais fácil se tivéssemos um mecanismo para criar um proxy mais dinâmico, semelhante ao fornecido pelo **SOAP::Lite**.  
No Capítulo 5, demonstraremos um proxy Java construído sobre o Apache SOAP que faz exatamente isso.

As operações no **Exemplo 4-24** seguem um padrão muito simples: inicializar a chamada SOAP, definir os parâmetros e invocar a chamada SOAP.

**Exemplo 4-24. Wrappers para as operações remotas**

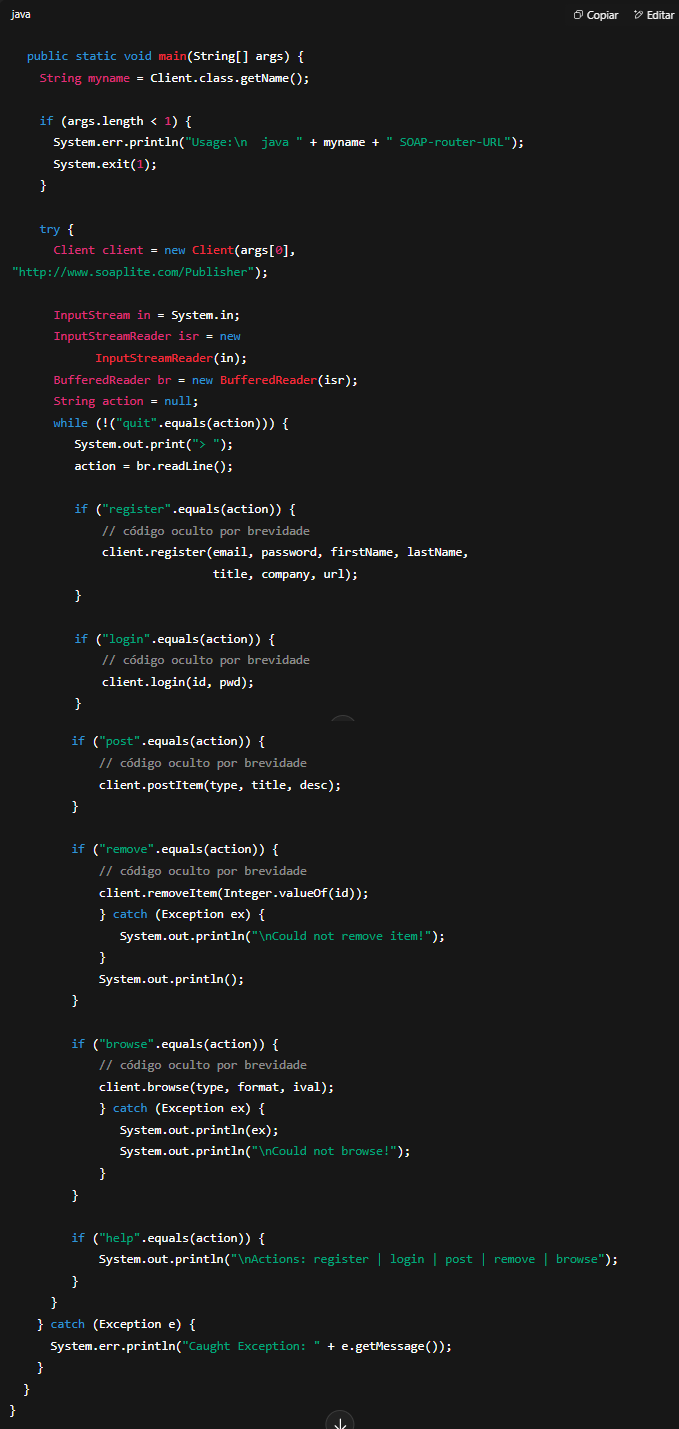


**4.4.2.5 O método main**

Agora que as operações básicas para interagir com o web service foram definidas, precisamos criar o código para o shell do Publisher (**Exemplo 4-25**).  
Este código não faz nada além de fornecer aos usuários um menu de opções para usar o serviço Publisher.  
Em um loop, obtemos a entrada do usuário, decidimos o que ele deseja fazer e executamos a ação.

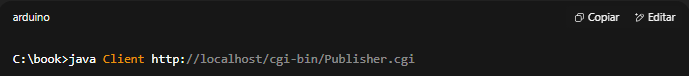
Como nenhum desse código lida diretamente com a invocação e uso do web service Publisher, partes significativas foram removidas por brevidade.  
O código completo pode ser encontrado no Apêndice C.

**Exemplo 4-25. O método main**



**4.4.3 Implantando o Cliente**

Depois que o código estiver escrito, compile-o e execute-o com o seguinte comando:



Substitua **localhost** pelo nome do servidor web onde o script CGI do Publisher está implantado.  
A **Figura 4-2** mostra o shell em execução.

**Figura 4-2. O shell Publisher em tempo de execução**

**

**Índice do capítulo:**

* **4. O Web Service Publisher**
* **4.1 Visão Geral**
* **4.2 As Operações do Publisher**
* **4.3 O Servidor Publisher**
* **4.4 O Cliente Shell em Java**